When Performance Portability is less than Perfect

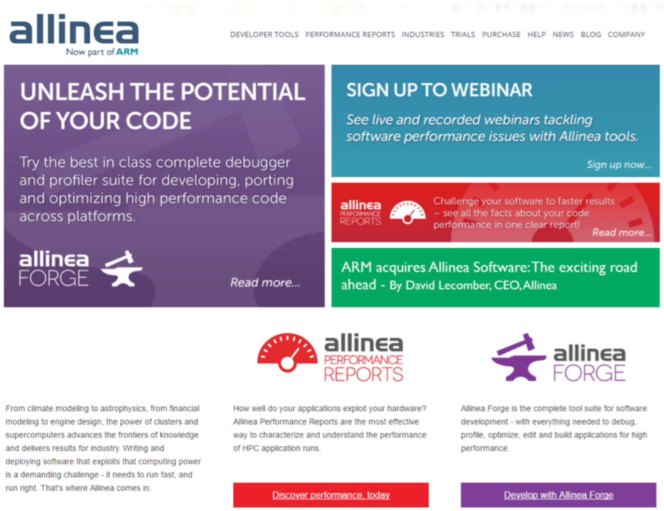
Matching Applications to Architectures

ARM

Mark O'Connor

Director Product Management, HPC & Server Tools

CoE PP, Denver
08 / 2017



allinea
Now part of ARM

DEVELOPER TOOLS PERFORMANCE REPORTS INDUSTRIES TRIALS PURCHASE HELP NEWS BLOG COMPANY

UNLEASH THE POTENTIAL OF YOUR CODE

Try the best in class complete debugger and profiler suite for developing, porting and optimizing high performance code across platforms.

allinea FORGE

Read more...

SIGN UP TO WEBINAR

See live and recorded webinars tackling software performance issues with Allinea tools.

Sign up now...

allinea PERFORMANCE REPORTS

Challenge your software to faster results – see all the facts about your code performance in one clear report!

Read more...

ARM acquires Allinea Software: The exciting road ahead - By David Lecomber, CEO, Allinea

allinea PERFORMANCE REPORTS

allinea FORGE

From climate modeling to astrophysics, from financial modeling to engine design, the power of clusters and supercomputers advances the frontiers of knowledge and delivers results for industry. Writing and deploying software that exploits that computing power is a demanding challenge - it needs to run fast, and run right. That's where Allinea comes in.

How well do your applications exploit your hardware? Allinea Performance Reports are the most effective way to characterize and understand the performance of HPC application runs.

Discover performance, today

Allinea Forge is the complete tool suite for software development - with everything needed to debug, profile, optimize, edit and build applications for high performance.

Develop with Allinea Forge

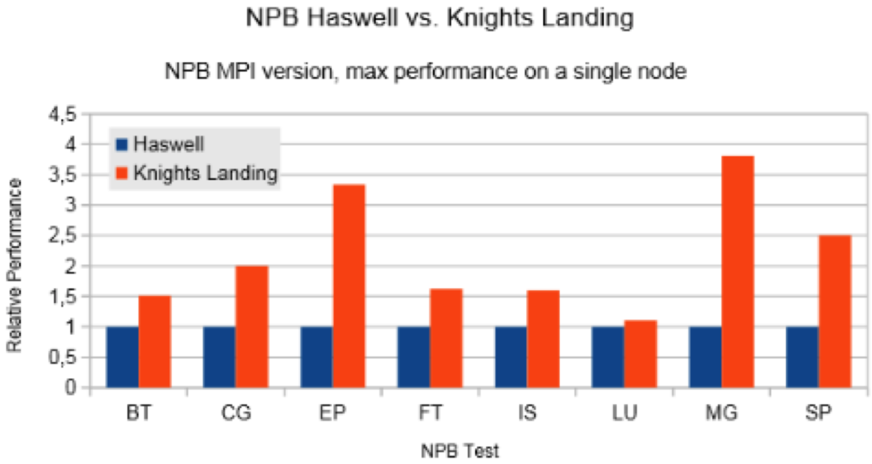
Allinea: leaders in cross-platform developer tools for HPC

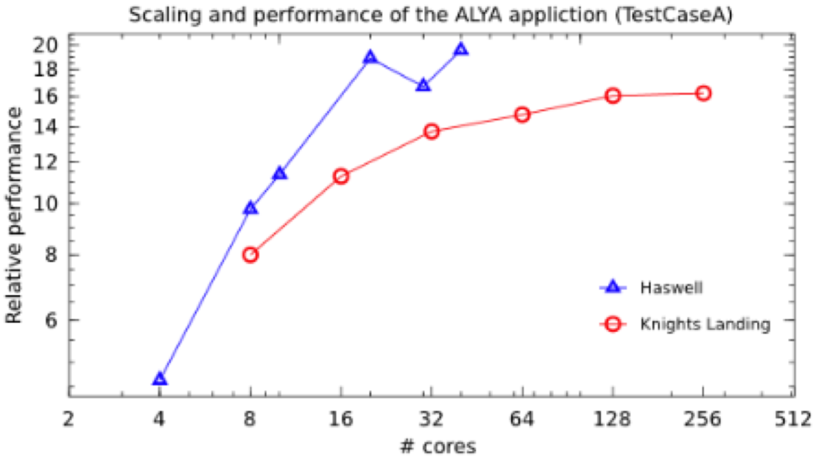
Los Alamos NATIONAL LABORATORY EST. 1943
Argonne NATIONAL LABORATORY
OAK RIDGE National Laboratory
NERSC NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER
University of California Lawrence Livermore National Laboratory
cea
HLRS
BAE SYSTEMS
TOTAL
BLUE WATERS SUSTAINED PETASCALE COMPUTING
TACC
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación
BERKELEY LAB
ICHEC Irish Centre for High-End Computing
GENCI
lrz
IFERC
ONERA THE FRENCH AEROSPACE LAB
SHARCNET™
archer
SNIC

Portable performance ≠ equivalent performance

Case study: Haswell vs KNL

Different applications benefit from the KNL architecture to different degrees



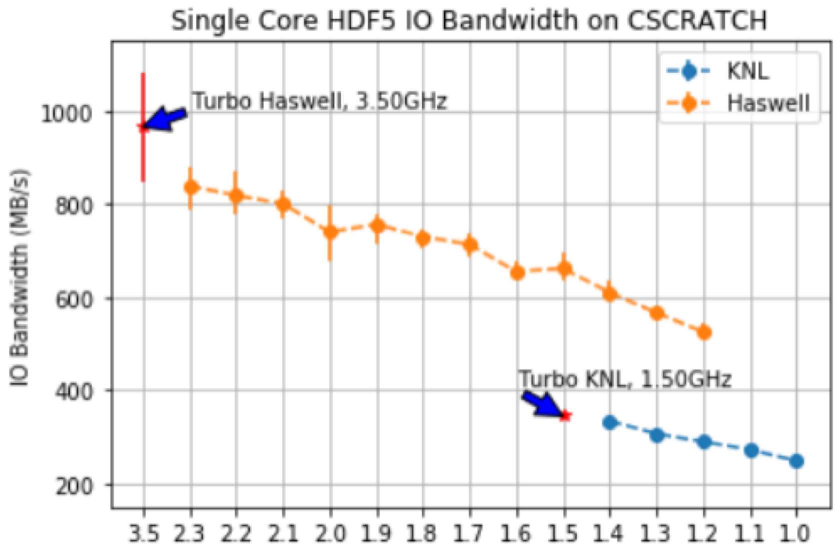


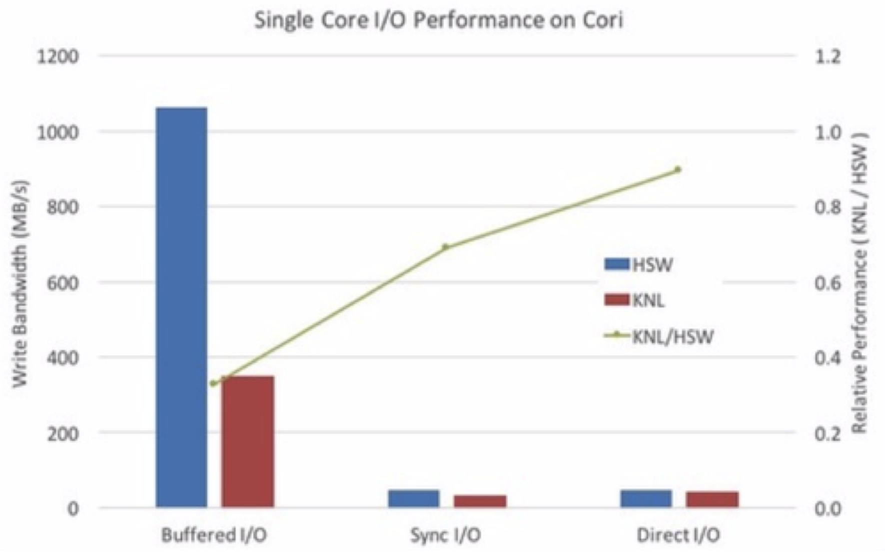
Source: http://www.prace-ri.eu/best-practice-guide-knights-landing-january-2017/

Can we predict application performance?

Cori saw I/O performance differences when changing CPU architecture

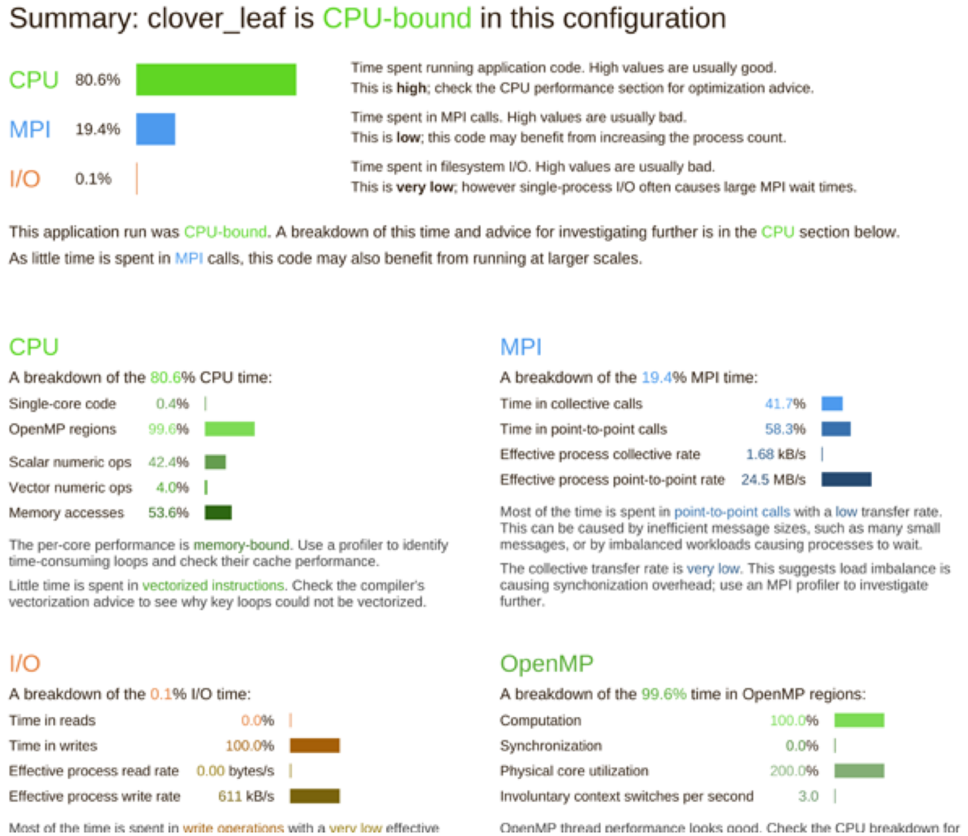
Microkernels work for single-core optimization, not application characterization





Source:
http://www.francoistessier.info/documents/CUG2017.pdf

Current attempts at application characterization



Summary: clover_leaf is CPU-bound in this configuration

CPU	80.6%	Time spent running application code. High values are usually good. This is **high**; check the CPU performance section for optimization advice.
MPI	19.4%	Time spent in MPI calls. High values are usually bad. This is **low**; this code may benefit from increasing the process count.
I/O	0.1%	Time spent in filesystem I/O. High values are usually bad. This is **very low**; however single-process I/O often causes large MPI wait times.

This application run was CPU-bound. A breakdown of this time and advice for investigating further is in the CPU section below.
As little time is spent in MPI calls, this code may also benefit from running at larger scales.

CPU

A breakdown of the 80.6% CPU time:

Single-core code	0.4%
OpenMP regions	99.6%
Scalar numeric ops	42.4%
Vector numeric ops	4.0%
Memory accesses	53.6%

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.

Little time is spent in vectorized instructions. Check the compiler's vectorization advice to see why key loops could not be vectorized.

MPI

A breakdown of the 19.4% MPI time:

Time in collective calls	41.7%
Time in point-to-point calls	58.3%
Effective process collective rate	1.68 kB/s
Effective process point-to-point rate	24.5 MB/s

Most of the time is spent in point-to-point calls with a low transfer rate. This can be caused by inefficient message sizes, such as many small messages, or by imbalanced workloads causing processes to wait.

The collective transfer rate is very low. This suggests load imbalance is causing synchonization overhead; use an MPI profiler to investigate further.

I/O

A breakdown of the 0.1% I/O time:

Time in reads	0.0%
Time in writes	100.0%
Effective process read rate	0.00 bytes/s
Effective process write rate	611 kB/s

Most of the time is spent in write operations with a very low effective

OpenMP

A breakdown of the 99.6% time in OpenMP regions:

Computation	100.0%
Synchronization	0.0%
Physical core utilization	200.0%
Involuntary context switches per second	3.0

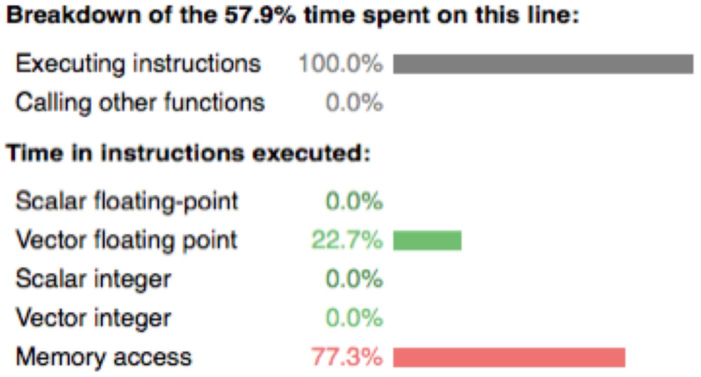
OpenMP thread performance looks good. Check the CPU breakdown for

Single page summary per application

- CPU vs MPI vs I/O breakdown
- Time in vector and memory instructions
- Effective MPI and I/O bandwidth
- Memory usage
- OpenMP / threading efficiency

Also available in Allinea MAP, plus:

- Breakdown over time and per source line:



Breakdown of the 57.9% time spent on this line:

Executing instructions	100.0%
Calling other functions	0.0%

Time in instructions executed:

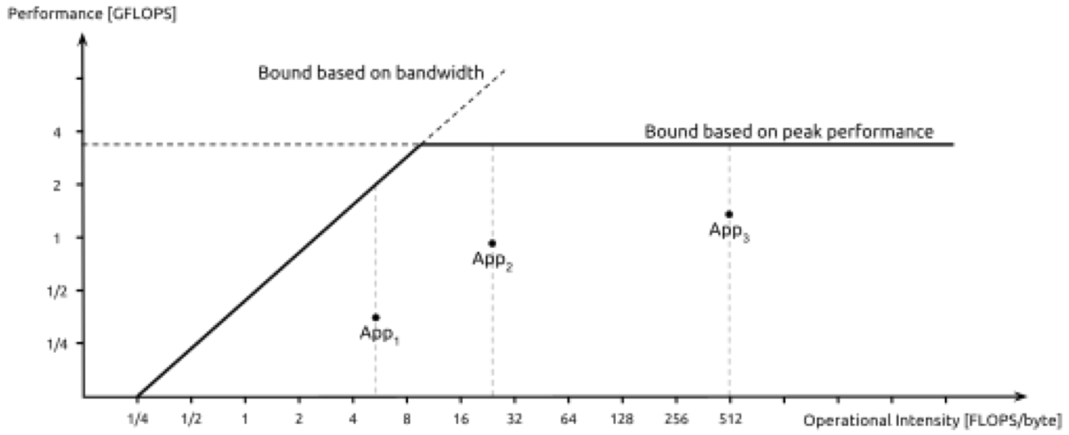
Scalar floating-point	0.0%
Vector floating point	22.7%
Scalar integer	0.0%
Vector integer	0.0%
Memory access	77.3%

Standardizing application characterization

Characterizing CPU performance – the roofline model

FLOPs/second plotted against FLOPs/byte – very successful for compute characteristics

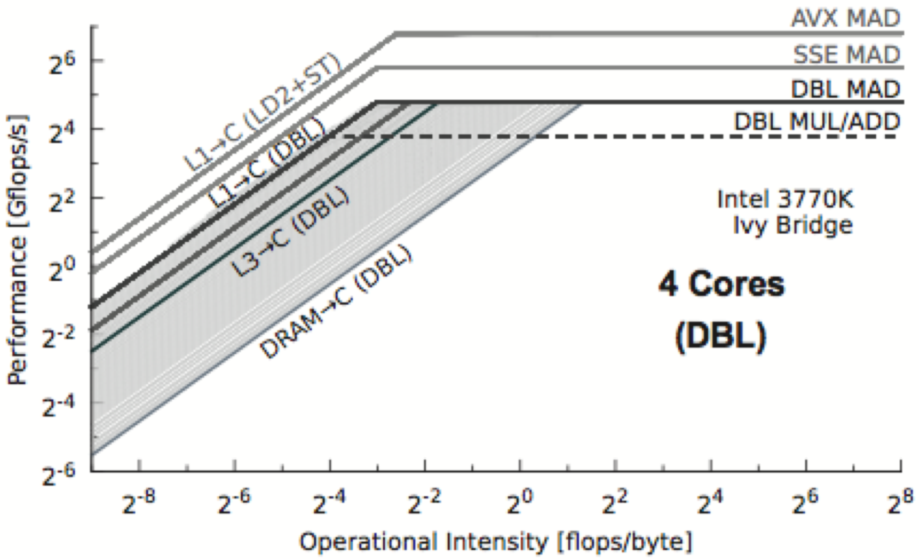
A CPU-centric view of application performance. Can we extend it for storage hierarchies?

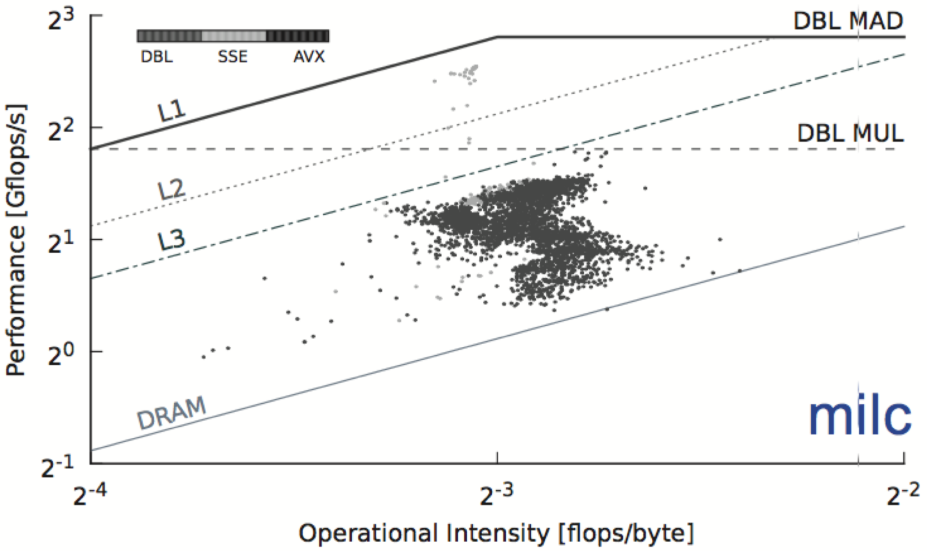


What would the roofline model for application portability look like?

Can we boil it down to two dimensions? Or a series of plots looking at different aspects?

Cache-aware roofline model





Extend to I/O-aware roofline model? Accelerator-aware roofline model?

If we want this, we need specific, reliable cross-manufacture performance counters.

Moving away from using as a loop-optimization tool to characterizing many applications on a cluster.

Source: http://sips.inesc-id.pt/~ilic/roofline.php

Exploring further extensions to the model

What happens if we tweak the vertical axis?

Performance isn't just Gflops/s. Some interesting alternatives:

Gflops/cycle

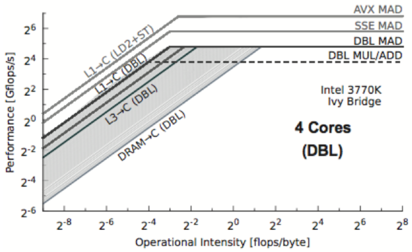
Does this allow us to compare different architectures more meaningfully?

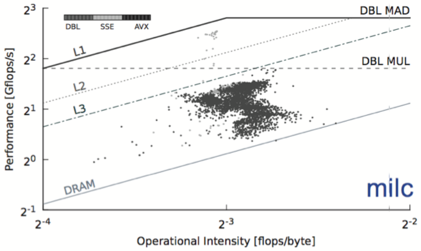
Gflops/joule

What if energy to solution across the cluster is more important than time to solution for each individual run?

Gflops/$

Can we now compare cloud offerings and bare-metal runs? Do our horizontal bars now show the trade-off in bursting to a cloud provider?





Other options to match applications to hardware

Machine learning

- Train an autoencoder to reduce dimensionality to 2D or 3D
- Cluster and predict

Reinforcement learning

- Train an end-to-end network to optimally place applications on clusters
- Plenty of training data available

Training and best practices

- Let *users* decide where to run their code
- Training humans may be harder than training machines

Thank-you!

Questions and discussion

ARM